

CAHIER DE RECHERCHES ET D'ETUDES N°34 : Framework et ORM



10 AVRIL

Créé par : DJIOJIP OUANKAP Claude Rowane

Pilote : P. MAXIMILLIEN BOSSOU

Promotion : X2026



Table des matières

| | |
|--|----|
| | 1 |
| Table des matières..... | 2 |
| I. CONTEXTE..... | 3 |
| II. ANALYSE DES BESOINS..... | 3 |
| III. PROBLEMATIQUE..... | 3 |
| IV. PLAN D’ACTION..... | 3 |
| V. REALISATION DU PLAN D’ACTION REEL..... | 4 |
| A. Définition des mots clés..... | 4 |
| B. Etude des framework PHP..... | 4 |
| D. Proposer un squelette de site utilisant les ORM..... | 9 |
| VI. VALIDATION DES HYPOTHÈSES..... | 12 |
| VII. CONCLUSION ET RETOUR SUR LES OBJECTIFS..... | 12 |
| VIII. BILAN CRITIQUE DU TRAVAIL EFFECTUE..... | 13 |
| IX. SYNTHÈSE DU TRAVAIL EFFECTUE ET DES RESULTATS OBTENUS..... | 13 |
| X. REFERENCES BIBLIOGRAPHIQUES FOURNIES DANS LE PROJET..... | 13 |
| XI. REFERENCES BIBLIOGRAPHIQUES COMPLÉMENTAIRES..... | 14 |

I. CONTEXTE

Suite aux multitudes appels d'offres perdus par l'agence, Julie et Marc évaluent le travail de l'entreprise et découvrent un réel problème de temps de développement ceci dû à la reprise de plusieurs éléments. Ils réfléchissent donc à comment créer un squelette de site disponible pour tous les sites et manipuler les données comme des objets.

II. ANALYSE DES BESOINS

- Optimiser le temps d'opération ;
- Créer un squelette de site ;
- Trouver un moyen de manipuler plus facilement les données.

III. PROBLEMATIQUE

Comment feront-ils pour créer un squelette de site et manipuler leurs données comme des objets ?

IV. PLAN D'ACTION

1. Définition des mots-clés ;
2. Étude des framework PHP ;
3. Étude des ORM ;
4. Proposer un squelette de site utilisant un ORM.

V. REALISATION DU PLAN D'ACTION REEL

A. Définition des mots clés

- **Workflow** : est la modélisation des processus métiers et de la gestion de ceux-ci. Encore appelé flux de travail, c'est la modélisation et la gestion des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier ou processus opérationnel.
- **Revue de code qualitatif** : est une opération consistant à analyser un code écrit afin de vérifier son fonctionnement.
- **Module de connexion** : est un bloc de code comportant les opérations de connexion à la base de données et pouvant être réutilisé dans plusieurs programmes.
- **Objet** : C'est une instance d'une classe.
- **Squelette de site** : C'est un template utilisé pour faciliter la conception d'un site contenant généralement une architecture prédéfinie.
- **PDO** : PHP Data Objects est une extension définissant l'interface pour accéder à une base de données avec PHP.

B. Etude des framework PHP

- **Définition** : C'est un cadre de travail, une boîte d'outils contenant des composants autonomes qui permettent de faciliter le développement d'un site web ou d'une application.
- **Objectifs/Utilité** : Le framework permet un gain de temps et d'efficacité pour le développeur, il lui sert de structure de base à chaque nouveau projet.

| <u>Avantages</u> | <u>Inconvénients</u> |
|--|---|
| Gain de temps (n'a pas besoin de tout redévelopper de A à Z) | Absence de maniabilité ou flexibilité : la structure étant déjà définie l'ajout d'une fonctionnalité complexe est difficile à implémenter |
| Maintenance simplifiée du code | Incertitude sur la capacité à maintenir un code dans le temps (vu qu'il dépendra de la solvabilité du framework) |
| Failles de sécurité sont préprotégées | Contrainte de poids, ses composants sont lourds |
| Avantage le travail en équipe (chacun contribue sur sa partie) | Le temps pour comprendre le fonctionnement du framework |

- Caractéristiques :

Les framework PHP ont les caractéristiques suivantes :

- _ Ils sont développés selon le design pattern MVC
- _ Ils sont structurés en POO (Programmation Orientée Objet)

- Fonctionnement :

Pour chaque service de l'application, il y a une structure correspondante, Un modèle qui gère l'accès à la base de données et est généralement codé en SQL, il permet le lancement des requêtes

Une vue qui gère l'affichage des données à partir des résultats des requêtes du modèle

Un contrôleur qui récupère les résultats du modèle et les transfère à la vue pour l'affichage

La particularité des framework est qu'il existe un super contrôleur appelé routeur, celui-ci reçoit une requête http et appelle le contrôleur correspondant pour le service.

- Différents frameworks :

Il existe plusieurs framework PHP à savoir principalement **Laravel** et **Symfony**

Installation de laravel :

After you have installed PHP and Composer, you may create a new Laravel project via the Composer `create-project` command:

```
composer create-project laravel/laravel example-app
```

Or, you may create new Laravel projects by globally installing the Laravel installer via Composer:

```
composer global require laravel/installer

laravel new example-app
```

After the project has been created, start Laravel's local development server using the Laravel's Artisan CLI `serve` command:

```
cd example-app

php artisan serve
```

Relier à une base de données :

Once you have configured your SQLite database, you may run your application's [database migrations](#), which will create your application's database tables:

```
php artisan migrate
```

- Alternatives Front-end :

Laravel étant un framework backend il peut être associé à plusieurs autres pour le front à savoir :

- React JS
- Vue JS
- Angular JS
- LiveWire (un framework php pour le front et compatible /créé pour Laravel)
- Ou Html, CSS avec Tailwind (framework css)

C. Etude des ORM

- Définition :

Object Relation Mapping est un ensemble de classe permettant de manipuler les tables d'une base de données relationnelle comme s'il s'agissait des objets. C'est une couche d'abstraction d'accès à la base de données qui donne l'impression/illusion de ne plus travailler avec des requêtes mais de manipuler des objets.

- Objectifs :

Il a pour but la réduction du code à écrire et à maintenir pour l'informaticien qui manipule la base de données depuis son logiciel, l'homogénéité du code objet et l'accélération du temps de développement.

| Avantages | Inconvénients |
|--|---|
| Eviter d'écrire du code SQL répétitifs | Ne peut appeler les fonctions ou les procédures prédéfinies et précompilées dans la base de données |
| Manipulation de la base de données à l'aide des objets | Empêche d'avoir les bases théoriques en gestion de base de données relationnelles |

| | |
|--|--|
| Accélération du temps de developpement | Ne peut effectuer des requêtes complexes : jointures, groupement, les transactions ou les traitements par lots |
| Possibilité de changer la base de données sans reprendre le code | L'ajout de cette couche logicielle rajoute des problèmes de maintenance et de performance |

- Caractéristiques :

Il sont caractérisés par des classes ayant des méthodes prédéfinies comme : create(), table(), etc

- Fonctionnement :

Il met à disposition des classes Objet pour manipuler les bases de données relationnelles ; le développeur manipule des objets et l'ORM transforme le tout en requêtes compréhensible par la base de données

- Différents ORM :

En PHP on distingue plusieurs ORM, parmi lesquels :

- Query Builder : fourni par Laravel :

Selects

Retrieving All Rows From A Table

```
$users = DB::table('users')->get();

foreach ($users as $user)
{
    var_dump($user->name);
}
```

Chunking Results From A Table

```
DB::table('users')->chunk(100, function($users)
{
    foreach ($users as $user)
    {
        //
    }
});
```

- Eloquent : fourni par Laravel

Generating Model Classes

To get started, let's create an Eloquent model. Models typically live in the `app\Models` directory and extend the `Illuminate\Database\Eloquent\Model` class. You may use the `make:model` [Artisan command](#) to generate a new model:

```
php artisan make:model Flight
```

If you would like to generate a [database migration](#) when you generate the model, you may use the `--migration` or `-m` option:

```
php artisan make:model Flight --migration
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Flight extends Model
{
    /**
     * The table associated with the model.
     *
     * @var string
     */
    protected $table = 'my_flights';
}
```

```
use App\Models\Flight;

foreach (Flight::all() as $flight) {
    echo $flight->name;
}
```

Building Queries

The Eloquent `all` method will return all of the results in the model's table. However, since each Eloquent model serves as a [query builder](#), you may add additional constraints to queries and then invoke the `get` method to retrieve the results:

```
$flights = Flight::where('active', 1)
    ->orderBy('name')
    ->take(10)
    ->get();
```

ORM

D. Proposer un squelette de site utilisant les ORM

Création de la base de données :

```
class CreateChirpsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('chirps', function (Blueprint $table) {
            $table->id();
            $table->foreignId('user_id')->constrained()->cascadeOnDelete();
            $table->string('message');
            $table->timestamps();
        });
    }
}
```

Utilisation ORM

```
class ChirpController extends Controller
{
  /**
   * Display a listing of the resource.
   *
   * @return View
   */
  public function index(): View
  {
    $Item = DB::table( table: 'chirps' )->get();
    $item = DB::table( table: 'products' )->get();
    return view( view: 'index', ['chirps' => $Item, 'users'=>$item]);
  }
}
```

Routeur :

```
Route::resource( name: 'chirps', controller: ChirpController::class )
->only(['index', 'store'])
->middleware(['auth', 'verified']);
```

Vue :

```
<article>
  Insérer un commentaire
  <div class="max-w-2xl mx-auto p-4 sm:p-6 lg:p-8">
    <form method="POST" action="{{ route('chirps.store') }}">
      @csrf
      <textarea
        name="message"
        placeholder="{{ __('What's on your mind?') }}"
        class="block w-full border-gray-300 focus:border-indigo-300 focus:ring focus:ring-indigo-200 focus:ring-opacity-50">
      >{{ old('message') }}</textarea>
      <button class="mt-4">{{ __('Chirp') }}</button>
    </form>
  </div>
</article>

<article id="commentaire">
  @foreach($chirps as $element)
    <div>
      <div class="card" style="width: 18rem;" id="item">
        <div class="card-body">
          <p class="card-text">{{ $element->message }}</p>
        </div>
      </div>
    </div>
  @endforeach
</article>
```

Modèle :

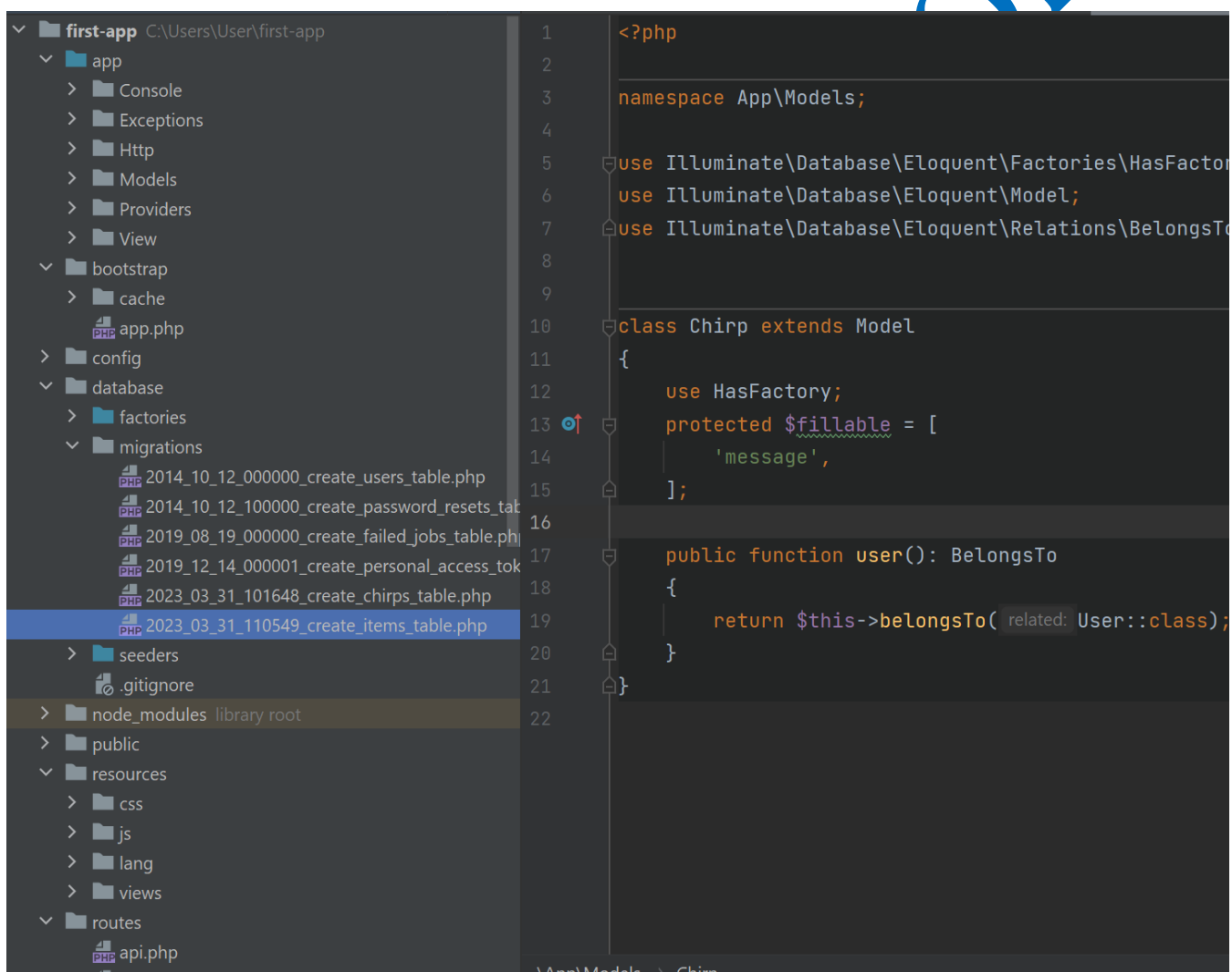
```

class Chirp extends Model
{
    use HasFactory;
    protected $fillable = [
        'message',
    ];

    public function user(): BelongsTo
    {
        return $this->belongsTo(related: User::class);
    }
}

```

Structure globale :



The screenshot displays a code editor with a file explorer on the left and a PHP code file on the right. The file explorer shows a project structure with folders like `app`, `bootstrap`, `config`, `database`, and `migrations`. The code editor shows the `Chirp` model class with its attributes and a `user` relationship method.

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Chirp extends Model
{
    use HasFactory;
    protected $fillable = [
        'message',
    ];

    public function user(): BelongsTo
    {
        return $this->belongsTo(related: User::class);
    }
}

```

Insérer un commentaire

What's on your mind?

Chirp

bonjour

bonjour

bye

avoir

reussi

VI. VALIDATION DES HYPOTHÈSES

Devons-nous étudier:

- Les framework PHP ?
- Les squelettes de site ?
- La manipulation des données comme objet ?

VII. CONCLUSION ET RETOUR SUR LES OBJECTIFS

Savoir en php :

[OBJ1] Décrit les avantages et le fonctionnement d'un Framework PHP

[OBJ2] Expérimente l'utilisation d'un Framework

[OBJ3] Explique les caractéristique d'un ORM

[OBJ4] Utiliser un ORM

VII. BILAN CRITIQUE DU TRAVAIL EFFECTUE

- **Individuel:** Ce prosit m'a permis de mieux assimiler les notions sur le développement Web, le design pattern MVC, les framework PHP et l'utilisation des ORM.
- **Collective:** Ce prosit a été bien travaillé, chacun a participé au workshop et à l'élaboration de la solution.

IX. SYNTHESE DU TRAVAIL EFFECTUE ET DES RESULTATS OBTENUS

Travail effectué :

1. Définition des mots-clés
2. Étude des framework PHP
3. Étude des ORM
4. Mise en place du squelette du site avec utilisation des ORM

Résultats obtenus :

- Squelette de site en Laravel connecté à une base de données et utilisant l'ORM Query Builder.

X. REFERENCES BIBLIOGRAPHIQUES FOURNIES DANS LE PROSIT

Pour la solution du Prosit:

- Workshop Laravel

XI. REFERENCES BIBLIOGRAPHIQUES COMPLEMENTAIRES

- [Google.](#)
- <https://bootcamp.laravel.com/blade/creating-chirps>
- <https://laravel.com/docs/10.x/installation>
- <https://laravel.com/docs/5.0/queries>
- <https://laravel.com/docs/10.x/eloquent>
- <https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/>

FRAMEWORK ET ORM